What is claimed is:

1. A method for producing advanced management information systems (MIS) information from source transaction data, comprising:

capturing source transaction data written sequentially in a pre-defined binary format from a database storing the data;

parsing the source transaction data to a plain text file format;

assigning a unique integer key value to each of a plurality of individual transaction records in each of a plurality of plain text output files; and

loading the output files to a relational database management system.

2. The method of claim 1, wherein capturing the source transaction data further comprises capturing the data written sequentially in a pre-defined binary elementized message format from the database.

3. The method of claim 1, wherein capturing the source transaction data further comprises capturing at least one of automatic teller machine (ATM) source transaction data and home banking system source transaction data from the database.

4. The method of claim 3, wherein capturing the ATM source transaction data further comprises capturing ATM transaction journal logs from the database.

5. The method of claim 3, wherein capturing the home banking system source transaction data further comprises capturing home banking system transaction journal logs from the database.

6. The method of claim 1, wherein capturing the source transaction data further comprises storing the source transaction data from at least one of an ATM system server and a home banking system server by the database.

7. The method of claim 6, wherein storing the ATM system source transaction data further comprises storing ATM transaction journal logs by the database.

8. The method of claim 7, wherein storing the ATM transaction journal logs further comprises decrypting the ATM transaction journal logs.

9. The method of claim 6, wherein storing the home banking system source transaction data further comprises storing home banking system transaction journal logs by the database.

10. The method of claim 9, wherein storing the home banking system transaction journal logs files further comprises decrypting the home banking system transaction journal logs.

11. The method of claim 1, wherein parsing the source transaction data further comprises parsing individual transaction records to ASCII text file format.

12 The method of claim 11, wherein parsing the individual transaction records further comprises converting all fields in the individual transaction records to ASCII text file format.

13. The method of claim 11, wherein parsing the individual transaction journal records further comprises producing columnized text for the individual transaction records.

14. The method of claim 11, wherein parsing the individual transaction journal records further comprises formatting dates and times of the individual transaction records into recognizable Structured Query Language (SQL) formatted values.

15. The method of claim 1, wherein parsing the source transaction data further comprises examining each of a plurality of transaction entries of the source transaction data to determine a type of function of each transaction corresponding to a transaction entry.

16. The method of claim 15, wherein parsing the source transaction data further comprises writing each transaction encountered in examining the transaction entries of the source transaction data as a line to an output file.

17. The method of claim 15, wherein parsing the source transaction data further comprises grouping all transactions encountered in examining the transaction entries of the source transaction data by transaction according to a data column of a transaction journal record that refers back to a session log record.

18. The method of claim 1, wherein assigning the unique integer key value further comprises assigning the unique integer key value to each individual transaction record in each of a plurality of ASCII text output files.

19. The method of claim 1, wherein assigning the unique key value further comprises writing the integer key value to a pre-defined column in each output file for loading to the relational database system.

20. The method of claim 19, wherein assigning the unique integer key value further comprises computing transaction times for each of the individual transaction records.

21. The method of claim 20, wherein computing the transaction times further comprises adding the computed transaction time to each output file.

22. A method for producing advanced management information systems (MIS) information from source transaction data, comprising:

reading at least one transaction journal log written in a pre-defined binary format to isolate a transaction message in a message buffer;

parsing the contents of the isolated transaction message;

writing out the parsed contents into a flat-text file of Structured Query Language (SQL) records into an output file loadable into a database re-using at least one Visual Basic (VB) class used for creating the transaction journal log; and

storing the output file in the database.

23. The method of claim 22, wherein reading the transaction log further comprises reading the transaction log for at least one financial institution application selected from a group consisting of an account information application, a bill payment application, a transfer application, a customer service channel report application, and a standards compliance application.

24. The method of claim 22, wherein reading the transaction journal log further comprises reading the transaction journal log written in a binary elementized message format.

25. The method of claim 22, wherein reading the transaction journal log further comprises reading a binary transaction journal log input file.

26. The method of claim 22, wherein reading the transaction journal log further comprises isolating the transaction message buffer.

27. The method of claim 26, wherein isolating the transaction message buffer further comprises determining at least one function code of the transaction message.

28. The method of claim 22, wherein reading the transaction journal log further comprises reusing at least one Visual Basic (VB) class that created the transaction message to re-create the contents of the transaction message from the transaction journal log.

29. The method of claim 28, wherein reusing the VB class to re-create the contents of the transaction message further comprises writing the transaction message in VB and via the VB class to produce the transaction journal log.

30. The method of claim 22, wherein reading the transaction journal log further comprises reading the transaction journal log by a VB application.

31. The method of claim 30, wherein reading the transaction journal log further comprises reading the transaction journal log by the VB application having a command line for a user interface for invocation.

32. The method of claim 31, wherein reading the transaction journal log further comprises reading the transaction journal log by the VB application which is started with the name of an input file given on the command line.

33. The method of claim 30, wherein reading the transaction journal log further comprises reading the transaction journal log by the VB application that includes via a VB object manager all known message type class-files that the VB application processes.

34. The method of claim 30, wherein reading the transaction journal log further comprises reading the transaction journal log by a VB object linking and embedding (OLE) control component (TJPARSE.OCX) of the VB application.

35. The method of claim 34, wherein reading the transaction journal log further comprises defining a VB form as a container for at least one VB OLE control component (MSGOCX.OCX and TJPARSE.OCX).

36. The method of claim 35, wherein defining the VB form as a container for the VB OLE control component further comprises defining the VB form that also displays status and offers a GUI to the VB application.

37. The method of claim 30, wherein reading the transaction journal log further comprises calling a method (TJPARSE.OCX) by the VB application to find a next transaction message in the transaction journal log.

38. The method of claim 37, wherein finding the next transaction message further comprises returning a status to the VB application indicating one of if an end-of-file was reached and if a garbled message was encountered.

39. The method of claim 37, wherein finding the next transaction message further comprises isolating the transaction message into a buffer space.

40. The method of claim 39, wherein isolating the transaction message further comprises calling a property of the called method (TJPARSE.OCX) to acquire an address of the message buffer space.

41. The method of claim 40, wherein isolating the transaction message further comprises acquiring the message buffer space address in the form of a VB Variant data type.

42. The method of claim 37, wherein reading the transaction journal log further comprises invoking an initialization Application Program Interface (API) by the called method for opening a transaction journal log file.

43. The method of claim 42, wherein invoking the initialization API further comprises performing an internal initialization.

44. The method of claim 43, wherein invoking the initialization API further comprises closing a previously opened file, if a file was previously opened.

45. The method of claim 42, wherein invoking the initialization API further comprises opening and initializing a new transaction journal log file.

46. The method of claim 37, wherein reading the transaction journal log further comprises invoking an isolating the next transaction journal message API by the called method for positioning the next transaction message in the transaction journal log, reading the transaction message contents, and inserting the transaction message in a static buffer.

47. The method of claim 46, wherein invoking the isolating the next transaction journal message API further comprises returning a status to the VB application selected from a group of statuses indicating at least one of a function code for the transaction message, if an end-of-file in the transaction journal log was reached, and if a garbled section of the transaction journal log was encountered.

48. The method of claim 47, wherein invoking the isolating the next transaction journal message API further comprises electing one of quitting and moving on to another transaction message, if the end-of-file status is returned.

49. The method of claim 47, wherein invoking the isolating the next transaction journal message API further comprises electing one of quitting and continuing to a next recognizable section of the transaction journal log, if the garbled section of the transaction journal log status is returned.

50. The method of claim 47, wherein invoking the isolating the next transaction journal message API further comprises returning a function code for the message, if the function code status is returned.

51. The method of claim 50, wherein invoking the isolating the next transaction journal message API further comprises using the returned message function code to determine a VB class to populate.

52. The method of claim 37, wherein reading the transaction journal log further comprises invoking a returning a transaction message buffer address API for returning the transaction message buffer address.

53. The method of claim 52, wherein returning the transaction message buffer address further comprises returning an internal message buffer memory address used by a message processing method.

54. The method of claim 22, wherein parsing the contents of the transaction message further comprises parsing the contents of the message into individual ELF variables by a VB application component (MSGOCX.OCX).

55. The method of claim 54, wherein parsing the contents of the transaction message further comprises parsing an ELF formatted binary transaction message buffer into VB variables by a message processing VB application component (MSGOCX.OCX).

56. The method of claim 54, wherein parsing the contents of the transaction message further comprises parsing the isolated transaction message buffer by a message processing VB application component consisting at least in part of a VB class file.

57. The method of claim 54, wherein parsing the contents of the transaction message further comprises parsing the isolated transaction message buffer by an API method via an application-supplied transaction message buffer address.

58. The method of claim 54, wherein parsing the contents of the transaction message further comprises parsing the isolated transaction message buffer into variables within a VB class file via an API to an internal subroutine.

59. The method of claim 58, wherein parsing the contents of the transaction message further comprises parsing the isolated transaction message buffer into VB class variables via the internal subroutine which is made public via an OLE property.

60. The method of claim 22, wherein writing out the parsed contents further comprises writing out parsed elementized message format contents into the flat text file of SQL records.

61. The method of claim 60, wherein writing out the parsed elementized message format contents into the flat text file of SQL records further comprises invoking an open elementized message format output file for opening the output file.

62. The method of claim 61, wherein writing out the parsed elementized message format contents into the flat text file of SQL records further comprises invoking a writing the elementized message format output file for writing out the parsed elementized message format contents on a pre-defined date range to the output file.

63. The method of claim 62, wherein writing out the elementized message format contents into the flat text file of SQL records further comprises invoking a closing the elementized message format output file for closing the elementized message format output file.

64. The method of claim 63, wherein writing out the elementized message format contents into the flat text file of SQL records further comprises starting another elementized message format output file.

65. The method of claim 22, wherein writing out the parsed contents further comprises finding a next transaction message in the transaction log.

66. The method of claim 65, wherein finding the next transaction message further comprises closing all input and output files, if an end-of-file is reached.

67. The method of claim 22, further comprising extracting a predefined date-range of transaction log files out of the transaction journal log based on a predetermined beginning and ending date.

68. The method of claim 67, wherein extracting the predefined date-range of transaction log files further comprises examining a VB date variable in a related VB class.

69. The method of claim 67, wherein extracting the predefined date-range of transaction log files further comprises writing out the pre-defined date-range of transaction log files to the output file.

70. The method of claim 22, further comprising running an audit check on the SQL records.

71. The method of claim 22, further comprising providing at least one type of report from the database selected from a group consisting of an overall summary session type of report, a statistics summary type of report, a functional usage summary type of report, a payment/transfer activity summary type of report, and a functional activity type of report.

72. The method of claim 22, further comprising providing at least one standard report from the database selected from a group consisting of a session summary report, a usage comparison report, a functions summary report, a customer activation /usage report, a session completion details report , an error reports, and a functions detail report.

73. A method for producing management information systems information from transaction journal log files written in elementized message format, comprising:

calling a method to find a next message in the transaction journal log files;

finding an end of a transaction journal log file and isolating message contents of the file into a message buffer space;

returning a status indicating the end of the file was reached;

calling another property of the method to acquire an address of the message buffer space;

providing the message buffer space address in the form of a VB VARIANT data type;

parsing the contents of the message corresponding to the message buffer address into elementized message format variables within at least one VB class;

performing at least one of writing a structured query language record of the parsed message contents into an output file and examining VB date variables in a predetermined VB class to determine if the parsed message contents should be written to an elementized message format output file; and

storing the output file in a relational database management system.

74. A system for producing advanced management information systems (MIS) information from source transaction data, comprising:

means for capturing source transaction data written sequentially in a pre-defined binary format from a database storing the data;

means for parsing the source transaction data to a plain text file format;

means for assigning a unique integer key value to each of a plurality of individual transaction records in each of a plurality of plain text output files; and

means for loading the output file to a relational database management system.

75. A system for producing advanced management information systems (MIS) information from source transaction data, comprising:

means for reading at least one transaction journal log written in a pre-defined binary format to isolate a transaction message in a message buffer;

means for parsing the contents of the isolated transaction message;

means for writing out the parsed contents into a flat-text file of Structured Query Language (SQL) records into an output file loadable into a database re-using at least one Visual Basic (VB) class used for creating the transaction journal log; and

means for storing the output file in the database.

76. A system for producing management information systems information from transaction journal log files written in elementized message format, comprising:

means for calling a method to find a next message in the transaction journal log files;

means for finding an end of a transaction journal log file and isolating message contents of the log file into a message buffer space;

means for returning a status indicating the end of the file was reached;

means for calling another property of the method to acquire an address of the message buffer space;

means for providing the message buffer space address in the form of a VB VARIANT data type;

means for parsing the contents of the message corresponding to the message buffer address into elementized message format variables within at least one VB class;

means for performing at least one of writing a structured query language record of the parsed message contents into an output file and examining VB date variables in

a predetermined VB class to determine if the parsed message contents should be written to an elementized message format output file; and

means for storing the output file in a relational database management system.